Pràctica 5: Disseny web

Aplicacions i serveis d'internet — Enginyeria de Sistemes TIC

Francisco del Águila López Aleix Llusà Serra

24 d'abril de 2023

Índex

Resum

Llenguatges web (WWW): HTML, CSS, JavaScript

1 Organització

1.1 Objectius

El objectius d'aquesta pràctica són:

1. Assolir nocions bàsiques del llenguatge HTML.
2. Entendre i aprendre a crear fulls d'estil.
3. Incorporar elements interactius a la pàgina web dissenyada, utilitzant JavaScript.
4. Aprendre a crear i a validar formularis en HTML5.

1.2 Condicions

- La pràctica està calibrada per a ésser treballada en equips de dues persones.
- La durada de la pràctica és d'1 setmana.

1.3 Lliurables

Heu d'entregar:

- Una carpeta comprimida on hi haurà tots els fitxers necessàris per poder provar la pàgina web dissenyada amb totes les seves característiques i funcionalitats.

1.4 Material necessari

Per dur a terme la pràctica cal tenir instal·lades eines de navegació web. En el cas de GNU/Debian i equiparables cal que instal·leu els paquets `firefox` o `firefox-esr`. També ens caldrà un editor de text per crear els nostres fitxers HTML. Així cal que, també, instal·leu el paquet `emacs`.

> Tasca prèvia 1 En aquesta pràctica no caldrà treballar en cap màquina virtual. Per tant, instal·leu els paquets anteriors si no els teniu, en el vostre computador.

2 Introducció al llenguatge HTML i CSS

L'HTML(HyperText Markup Language) és un llenguatge que permet descriure hipertext, és a dir, text presentat de forma estructurada per mitjà de paraules clau. En principi direm que HTML no és un llenguatge de programació sinó que és un llenguatge d'etiquetes(tags) que comuniquen al navegador quina és l'informació a mostrar per pantalla i el seu format. En particular, l'enfocament més modern d'HTML implica una descripció semàntica del contigut de la web i deixa per altres eines com CSS la descripció d'estil del contingut de la web.

2.1 Versions d'HTML

Des de la creació de l'HTML hi hagut molts canvis en el propi llenguatge. Inclús, ha hagut una lluita entre consorcis d'entitats com W3C i WHATWG per vetllar per l'existència d'un únic estàndard obert que absorbís les demandes evolutives de l'estàndard. Actualment la versió de

l'estàndard és *HTML Living Standard* governades per *WHATWG* (Web Hypertext Application Technology Working Group) [Web23]. Podeu trobar informació de l'estàndard i informació històrica a [Wik23].

2.2 Estructura bàsica d'una pàgina web HTML

L'esquelet bàsic d'una pàgina web dissenyada amb HTML és la següent:

```
<!DOCTYPE HTML>
<html lang="ca">
  <head>
    <meta charset="UTF-8">
    <title>Títol de la pàgina web</title>
  </head>
  <body>
     <p>Cos de la pàgina</p>
  </body>
</html>
```

En la primera línia es troba una etiqueta relativa a la versió/variant de HTML. En realitat no és necessària, però indica l'estàndard que utilitzem i ens servirà per validar el codi.
A continuació hi ha les etiquetes que indiquen que el text és html. Totes les etiquetes s'han d'obrir i tancar. Tot el contingut que hi ha entremig, pertany en aquest bloc(el quin marca l'etiqueta). La capçalera és la secció compresa entre les etiquetes de head. En ella es troba el títol(entre les etiquetes title). Aquest títol no apareix a la pàgina en si, sinó que és el text que apareix a dalt la finestra del navegador. També conté l'etiqueta meta amb l'atribut charset que indica el tipus de codificació del fitxer on hi ha el contingut HTML.
El cos de la pàgina està delimitat per les etiquetes body.
És important dir que les etiquetes poden tenir atributs. Els atributs són una espècie d'opcions que s'apliquen a l'element (etiqueta). L'atribut de l'etiqueta html és lang, que indica el llenguatge que es fa servir al document. En el cas del cos, per exemple podríem tenir:

```
<body style="background-color:blue;color:green;background-image:url(‘‘fitxer_local.gif’’)"
dir="rtl">
```

Aquests atributs són:

- style: serveix per definir l'estil del body. Consta de diferents sub-atributs.
  - background−color: serveix per especificar un color pel fons de la pàgina. El valor d'aquest atribut és pot posar amb els noms en anglés dels colors o en el format: #rrggbb. És a dir el caràcter # seguit de 6 lletres o números en l'ordre: Red, Green, Blue.
  - color: serveix per especificar el color del text. És el mateix format que el background-color.
  - background−image: ens permet especificar una imatge pel fons de la pàgina.
- dir: Indica en quina direcció anirà el text.

Un dels millors llocs de referència per aprendre HTML i altres tecnologies relacionades amb la web és [W3S23c]. Torbareu tutorials, la llista referenciada de tags HTML, etc.

2.3 Etiquetes bàsiques

Comentaris en el codi Els comentaris els podem posar de la següent forma: <!−−*Comentari* −−>

Paràgrafs Per definir un nou paràgraf s'utilitzen les etiquetes següents: <**p**>El text del paràgraf </**p**>

Salt de línia. Un salt de línia en HTML s'escriu com a: <**br**>

Formats de text: negreta i cursiva Per escriure un text en negreta, s'ha d'incloure entre les etiquetes: <**b**> i </**b**>. En el cas de la cursiva, les etiquetes són: <**i**>i </**i**>.

Enllaços Per incloure enllaços a altres pàgines web o fitxers, tan externs com locals, s'utilitza l'etiqueta a. El format és el següent: <**a href**="ruta_objecte_web">nom_del_enllaç</**a** >. On el nom_del_enllaç serà el nom del link que es veurà a la pàgina web.

Llistes Per crear llistes no ordenades, és a dir, sense enumeració, es fa de la següent forma:

```
<ul>
  <li> Un element.</li>
  <li> Un altre element.</li>
</ul>
<ul style="list-style-type:square">
  <li> Un element de llista amb marcador quadrat</li>
  <li> Un altre element.</li>
</ul>
```

I si volem crear llistes enumerades és d'aquesta forma:

```
<ol>
  <li value="3"> Aquesta llista comença amb valor 3.</li>
  <li> Aquest serà l'element 4.</li>
  <li> Aquest el 5.</li>
</ol>
```

Imatges Per incloure imatges es fa de la següent forma: <**img src**="ruta_imatge"**alt**="Imatge 1"> . En aquest exemple es veu l'atribut alt. Aquest atribut serveix per especificar el text que es mostrarà si l'imatge no es pot carregar, o mentre es carrega o un cop carregada, es passa el ratolí per sobre.
Hi ha altres atributs que es poden afegir que són interessants, com: width que especifica l'amplada de l'imatge en píxels. height que serà l'altura de l'imatge. Aquests dos últims atributs actualment es consideren estils. Per tant, seran valors de l'atribut style.

2.4 Fulls d'estil CSS

CSS és un llenguatge utilitzat en la presentació de documents HTML. La filosofia de CSS es basa en intentar separar lo que es l'estructura i la semàntica de la pàgina web de la seva presentació. És a dir, l'idea és tenir un fitxer de text HTML on hi hagi el contingut semàntic de la web i un fitxer .css que descrigui la forma en la qual es presenta aquest contingut a la web.
Com ja s'ha vist a l'apartat anterior, s'ha definit algun estil en HTML. Les bones pràctiques de desenvolupament Web condueixen a que tots els aspectes referents a l'estil de presentació estiguin en un fitxer d'estil CSS. Així el contingut del fitxer HTML només s'encarrega dels aspectes referents a l'estructura i semàntica del contingut. Això només és una petita introducció

dels fitxers d'estil, ja que poden arribar a ser molt complexes i potents.
Primer de tot, en la pàgina web s'ha d'importar el full d'estil que es vol aplicar. Per importar-lo s'ha de fer dins de la capçalera (head) de la següent forma: <**link rel**="stylesheet"**type**="text/css"**href**="fitxer.css">
Per entendre com es defineixen els estils amb CSS, anem a veure un exemple:
H1 {color:black; font−family:arial; font−size:20pt; text−align:center}

- h1 és l'etiqueta a la qual s'apliquen les característiques que es defineixen a continuació. Aquesta etiqueta ha d'existir en HTML i tot el text que escrivim dins d'aquesta etiqueta tindrà el format que especifiquem ara, en el full d'estil .css.
  Abans no s'ha comentat, però les etiquetes d'HTML: h1,h2,h3,... serveixen per establir diferents nivells de capçaleres. Ara nosaltres, en el full d'estil podrem especificar per cada etiqueta quin format volem.
- color ens especifica el color de la lletra.
- font−family ens especifica l'estil de lletra. Els possibles estils de lletra en realitat dependran dels quins tinguem instal·lats al nostre sistema.
- font−size ens especifica la mida de la lletra.
- text−align ens permet posicionar el text: left, center, right, middle, top i bottom.

2.5 Eines de desenvolupament

Per fer el desenvolupament de les nostres pàgines web, utilitzarem 3 eines diferents:

1. L'editor de text Emacs. Tots els fitxers HTML i css els crearem amb l'emacs, editor que tots vosaltres ja coneixeu.
2. Un validador online d'HTML `http://validator.w3.org/#validate_by_upload`.
   Per assegurar que les pàgines dissenyades, compleixen els estàndards marcats pel WHATWG, validarem els fitxers a través de la web anterior. Aquesta validació és molt important per assegurar que la nostre web es veurà correctament a tots els navegadors.
3. Eines de desenvolupament del Firefox. El propi navegador Firefox té unes eines pensades pels desenvolupadors de pàgines web. Aneu a *Més Eines* de Firefox i trobareu una opció anomenada: *Eines per Desenvolupador Web*, que podeu activar. Per fer-ho també podeu prémer les tecles:*Ctrl+Maj+I*.
   Per una banda tenim la consola web que ens mostrarà possibles errors que hi ha a la nostre pàgina. Una eina molt interessant es l'Inspector, que ens va mostrant quin fragment de codi HTML correspont a l'àrea per on passem el ratolí. Pot ser interessant per veure les nostres pàgines, però també per veure com estan dissenyades altres pàgines web que ens agradin.

Per ajudar-nos a crear les nostres pàgines web, fulls d'estil, els scripts de JavaScript, etc. consultarem la web (citada ja anteriorment): `http://www.w3schools.com/`.
En aquesta web podem trobar tota l'informació necessària sobre les etiquetes d'HTML, exemples de web o d'scripts i molta més informació.

2.6 Disseny bàsic de la pàgina web

Un cop adquirides les nocions bàsiques sobre el llenguatge, ja podem fer el nostre primer disseny de la pàgina web. Aquesta pàgina ha de tenir els elements següents:

1. Missatge de benvinguda.
2. Un apartat amb les dades del grup que ha dissenyat la web.
3. Un apartat amb enllaços a pàgines web favorites.
4. Un enllaç a una altra pàgina web (també vostre) que contingui fotografies.

Tasca 2 Dissenyeu dues pàgines web. Una serà la principal i ha de tenir els elements anteriors. I l'altre serà una pàgina web que contindrà fotografies.

Tasca 3 Definiu-vos un únic fitxer de text amb extensió .css que serveixi d'estil per a tots els documents HTML que formen part del vostre lloc web.

3 Funcionalitats interactives: JavaScript i HTML

JavaScript és un llenguatge de programació interpretat que permet crear petits programes per realitzar accions dins de l'àmbit d'una pàgina web. Amb aquests programes podem crear efectes especials en les pàgines i interactuar amb l'usuari visitant de la web. El navegador web és l'encarregat d'interpretar les instruccions de Javascript i executar-les per realitzar aquests efectes o interaccions.
Javascript és un llenguatge amb moltes possibilitats, permet la programació de petits scripts com serà el nostre cas, però també permet fer programes més grans, orientats a objectes, estructures de dades complexes, etc.

On s'escriu el codi JavaScript? La programació de JavaScript es realitza dins del propi document HTML. Per tant, en el codi font de la pàgina web hi tindrem els nostres scripts juntament amb la resta del codi HTML. Per fer-ho s'han d'incloure unes etiquetes HTML per delimitar l'script. Aquestes etiquetes són: <**script**> i </**script**>.

Sintaxi del llenguatge El llenguatge JavaScript té una sintaxi molt semblant a la de Java (ja que està basat en ell) i també molt semblant al C.
Per afegir comentaris dins del codi s'utilitza la doble barra: // o per fer comentaris de varies línies s'utilitza el signe /* per començar i el */ per acabar.
En aquest llenguatge s'han de respectar les majúscules i les minúscules.
Per separar instruccions s'utilitza el caràcter ;.

Com s'executen els scripts? Els scripts de JavaScript s'executen en resposta a esdeveniments. Els esdeveniments es poden produir de forma automàtica com per exemple *quan es carrega la pàgina* o bé correspondre a accions que realitza l'usuari. La programació amb JavaScript està enfocada a interceptar aquests esdeveniments i realitzar accions en resposta a aquests. Existeixen molts tipus d'esdeveniment: onmouseover, onmouseout, onclick, etc.
En aquesta pràctica s'utilitzarà l'esdeveniment oninput per quan l'usuari estigui situat sobre el camp de confirmar la contrasenya. Per fer ús d'esdeveniments en HTML s'han d'afegir com atributs: <etiqueta_HTML ... esdeveniment=instrucció> . On *instrucció* pot ser la crida a una funció creada amb JavaScript.

Per tant l'idea és que quan l'usuari estigui escrivint en el camp de confirmació de password, es cridi una funció que comprovi si aquell camp és igual al del password.

Funcions Les funcions en JavaScript es declaren com: function nom_funcio(paràmetres){ instruccions de la funció}. Hem d'estar alerta on escriure la declaració ja que s'ha de fer abans d'on estigui la crida.

Variables La declaració de variables en JavaScript està descrit a [W3S23d]. El mecanisme modern fa servir les paraules reservades *let* i *const*. Però es mantenen els altres mecanismes per qüestions històriques.

Objectes de JavaScript JavaScript treballa de forma natural amb Objectes. Normalment les variables definides a JavaScript es tracten com a Objectes. Però un dels aspectes més importants de JavaScript és el DOM (Document Object Model). Per mitjà del DOM, JavaScript pot accedir i canviar tots els elements d'un document HTML. Aquests elements estan mapats com un arbre d'objectes. Per a més informació consulteu [W3S23b].
Per fer que un script mostri algun missatge escrit a la pàgina, es pot utilitzar el mètode write que pertany a l'objecte document. Per tant, si fem: document.write('El text que volem escriure') podrem escriure a la pàgina el que vulguem.
Si en canvi, volem fer sortir un missatge emergent d'alerta, tenim el mètode alert() que pertany a l'objecte window, i només cal fer: alert('El missatge').
Per demanar una entrada a l'usuari de forma emergent, tenim el mètode prompt().

3.1 Formularis amb HTML i JavaScript

Els formularis són una característica de l'estàndard HTML. Permeten als autors recollir informació dels visitants de la web. Aquests formularis poden resultar útils per reunir informació personal, de contacte, opinions, etc.
La validació de formularis serveix per comprovar que les dades que ha entrat l'usuari són en el format correcte i compleixen les especificacions que es demanen i/o que no hi hagi camps en blanc (quan són camps obligatoris). Aquestes validacions han sigut un problema que sempre s'ha solucionat amb l'utilització de Javascript, ja sigui amb codi propi o llibreries creades per tercers. Però amb les versions actuals d'HTML es poden validar molts camps del formulari sense fer ús de cap script.
En aquesta pràctica dissenyarem un formulari i farem la seva validació amb HTML i utilitzarem JavaScript només per validar si dos camps són iguals.
A continuació es mostra un formulari com a exemple:

```
<!DOCTYPE HTML>
<html lang="ca">
  <head>
    <meta charset="UTF-8">
    <TITLE>Formulari amb HTML</TITLE>
    <script>
      function check() {
     if (document.getElementById('correu').value != document.getElementById('rep_correu').valu
         document.getElementById('rep_correu').setCustomValidity('El correu ha de ser el mate
         return false;
     } else {
         return true;
     }
```

```
      }
    </script>
  </head>
  <body>
    <form id="FORMULARI" action="" method="post" onsubmit="return␣check()">
      <br>
      <label for="nom_usuari">Nom d'usuari: </label>
      <input name="nom_usuari" id="nom_usuari" type="text" required>
      <br>
      <label for="correu">Correu òelectrnic: </label>
      <input name="correu" id="correu" type="email" required>
      <br>
      <label for="rep_correu">Repetir correu òelectrnic: </label>
      <input name="rep_correu" id="rep_correu" type="email">
      <br>
      <label for="provincia">íProvncia: </label>
      <input name="provincia" id="provincia" list="prov" type="text">
      <datalist id="prov">
        <option value="Barcelona">
        <option value="Girona">
        <option value="Lleida">
        <option value="Tarragona" >
      </datalist>
      <br>
      <br>
      <input type="submit" value="Enviar" id="boto_enviar">
      <input type="reset" value="Esborrar" id="boto_reset">
    </form>
  </body>
</html>
```

En aquest formulari tenim un script que comprova que el correu electrònic és el mateix que el del camp de confirmació del correu. Cal fer notar que amb aquest codi JavaScript queda sobrecarregat el mecanisme de validació que tenen els formularis en HTML:

- La crida del codi JavaSacript es realitza en l'esdeveniment onsubmit. La validació que ve incorporada també es realitza (sense JavaSacript) quan es produeix aquest esdeveniment.
- L'acció de *submit* es realitza si la validació és True. En cas contrari, es presenta el missatge corresponent a la propietat *validationMessage* de l'element de tipus input que no passa la validació i per tant, no es realitza l'acció de *submit*.
- el mètode *setCustomValidity()* serveix per modificar el *validationMessage* de l'element de tipus input.
- Per sobrecarregar la validació cal que la funció *check()* retorni un valor True o False i que aquest valor sigui lliurat a l'esdeveniment onsubmit per realitzar l'enviament o no del formulari. Aquest traspàs de condicions booleanes s'ha de realitzar independentment de les accions que faci la funció *check()*. Per aquest motiu la sintaxis que es fa servir és onsubmit="`return check()`" i no simplement onsubmit="`check()`".

Pel que fa al cos del formulari, al obrir l'etiqueta de form hi han els possibles atributs. L'atribut id simplement identifica el formulari dins del text HTML. L'action és l'acció que es durà a terme quan enviem el formulari i el method serveix per especificar com serà enviada l'informació del formulari. En aquesta pràctica, podeu deixar aquests dos atributs sense cap valor, ja que a la pròxima pràctica els omplirem correctament.

Per a cada camp per omplir contingut per part de l'usuari que vulguem crear i mostrar a la web, haurem d'especificar el text que volem que es mostri davant de la casella per omplir. Això s'aconsegueix amb l'etiqueta label. Observeu l'atribut for de l'element label. Hi ha diferents elements per fer entrades de formulari: input, button, checkbox, datalist, textarea, etc. A continuació es descriuen els que utilitzarem en aquesta pràctica:

- L'input permet l'entrada de molts tipus de dades de l'usuari. Amb l'atribut type s'especifica el tipus de dada i el navegador ho mostrarà d'una forma o una altre. Uns dels valors possibles per aquest atribut són: text, number, email, password, search, url, tel, date, submit, reset, etc.
  Segons el type, l'HTML comprova si el contingut és correcte. En els casos dels Smartphones o Tablets amb navegadors compatibles amb HMTL farà que en cada cas mostri el teclat que toca (per exemple si és de tipus number, ens mostrarà directament el teclat numèric).
  L'atribut required serveix per indicar que aquella entrada és obligatòria. No cal especificar cap valor, simplement incorporar-lo a la llista d'atributs de l'element.
  Un altre atribut interessant és el pattern que serveix per especificar limitacions més concretes de l'entrada d'aquell camp. Aquest atribut el podeu utilitzar voluntariament en aquesta pràctica.
  Si volem escriure un valor inicial en un dels camps o per exemple escriure un nom sobre d'un botó, utilitzarem l'atribut value.
  Cal fer notar la diferència entre l'atribut id i l'atribut name de l'element input. En aquest cas, id serveix per distingir i identificar un input d'un altre per part del JavaScript (cantó client). L'atribut name servirà per referenciar la informació que rebrà el servidor quan el formulari sigui enviat. Consulteu la documentació de referència per més detalls [W3S23c].

- El datalist serveix per oferir una llista de possibles respostes. L'estructura és:

```
<datalist id="xx">
    <option value="a" >
    <option value="b" >
</datalist>
```

  Per relacionar aquesta llista a l'entrada d'un camp, s'ha de crear un input corresponent i afegir com atribut: *list="xx"*, on el valor, és l'id del datalist.

- Una textarea és una zona on s'hi poden deixar comentaris. Té la mateixa estructura que un input, però canviant el nom de l'etiqueta. Els atributs interessants d'aquest camp són rows i cols que serviran per limitar el número de files i columnes de la zona de text.

3.2 Asynchronous JavaScript And XML (AJAX)

AJAX és un mecanisme per aconseguir que tant client com servidor puguin intercanviar informació després de carregar el contingut de la pàgina. Així, és possible actualitzar el seu contingut sense haver de recarregar la pàgina. També fa possible aquest intercanvi d'informació sense que intervingui l'usuari.

La clau d'aquest comportament es basa en la creació d'objectes del tipus XMLHttpRequest. Trobareu més informació a [W3S23a].

Observeu el codi següent:

```
<!DOCTYPE HTML>
```

```
<html lang="ca">
  <head>
    <meta charset="UTF-8">
    <title>Prova AJAX</title>

    <script>
      function loadDoc() {
     const xmlhttp;
     const objecte_div;
     objecte_div=document.getElementById("myDiv");
     xmlhttp=new XMLHttpRequest();
     xmlhttp.onload=function() {
         objecte_div.innerHTML=xmlhttp.responseText;
         objecte_div.style.color="green";
     }
     xmlhttp.open("GET","entrada.txt");
     xmlhttp.send();
      }
    </script>

  </head>
  <body>
    <div id="myDiv" style="color:blue"> Hola!!!!!! </div>

    <script>
    <!--Cada 3 segons, crida la ófunci loadXMLDoc()-->
    setInterval("loadDoc()",3000);
    </script>

  </body>
</html>
```

En aquest programa de JavaScript podem veure dues funcionalitats més, d'aquest llenguatge:

- Per una banda, al final del codi podeu veure un petit script, amb la crida a una funció predefinida setInterval. Aquesta funció serveix per temporitzar una acció. En aquest cas, cada 3000ms cridarà una funció anomenada loadDoc().

- Per altre banda, a la definició de la funció loadDoc(), tenim la creació d'un objecte XMLHttpRequest de nom xmlhttp. A la propietat xmlhttp.onload es defineix la funció que es vol executar quan es llenci la petició HTTP a través del mètode xmlhttp.send(). Es pot considerar que a xmlhttp.onload s'està definint el *Callback* que s'executarà quan es rebi la resposta de la petició HTTP. Amb el mètode xmlhttp.open() s'especifica com serà el missatge de petició HTTP.
  Cal fer notar que el recurs de tipus fitxer *entrada.txt* es troba localitzat al servidor. Per aquest motiu cal realitzar la petició HTTP. Observeu que si aquest fitxer canvia el contingut, també ho farà el contingut HTML que es visualitza en el navegador client sense cap acció per part de l'usuari. Així, tant el contingut com el color de l'element div amb id='myDiv' canviarà cada 3 segons en funció del contingut d'*entrada.txt*.
  Per temes de seguretat els navegadors actuals no permeten peticions HTTP amb objectes XMLHttpRequest que no siguin del mateix domini d'on s'ha descarregat el contingut HTML. A efectes pràctics implica que les peticions HTTP s'han de fer al mateix servidor. Un altre aspecte relacionat també amb la seguretat és que l'execució AJAX (JavaScript) de forma local (sense servidor) també està restringida de manera que no es pugui accedir a

fitxers de la màquina local on s'executa el navegador. Per aquest motiu, per provar aquest codi us haureu d'esperar a muntar el servidor web de la pràctica següent, i col·locar el fitxer *entrada.txt* en el sistema de fitxers del servidor per a que pugui ser servit com un fitxer més, de la mateixa manera com servirà els fitxers HTML.

TASCA PRÈVIA 4 Per provar aquest codi, haureu de crear un fitxer de text anomenat: *entrada.txt* en el directori del servidor on tingueu la pàgina web amb l'script anterior.

Considereu que aquest fitxer simplement conté un 0 o un 1. Modifiqueu l'script anterior per fer que si hi ha un 0 en el fitxer, el contingut de la pàgina es mostri de color blau i si hi ha un 1, de color verd. Ara el text sempre serà *Hola!!!!!!*, només ha de canviar el color.

Afegiu l'enllaç d'aquesta nova pàgina a la vostra pàgina principal. Deixeu aquesta tasca preparada per provar-la a la pràctica següent quan es tingui un servidor web operatiu.

4 Disseny d'un formulari

Ara en la vostra pàgina principal hi afegireu un nou enllaç: *Sol·licitud d'amistat*. Aquest enllaç ens portarà a una altre pàgina on hi haurà un formulari per a la introducció de les dades de la persona que visita la vostre pàgina. El formulari ha de permetre introduir les següents dades:

- Nom d'usuari
- Password
- Confirmació de password
- Sexe: escollir entre home i dona *(datalist)*
- Correu electrònic
- Comentaris, mitjançant una àrea de text
- Botons *Enviar* i *Esborrar*

Incorporeu el següent conjunt de validacions al formulari de la vostre pàgina personal:

1. Els camps *Nom d'usuari*, *Password* i *Correu electrònic* no poden estar en blanc.
2. S'ha de verificar que *Confirmació de password* coincideix amb *Password*.
3. S'ha de verificar que el *Correu electrònic* té el format correcte.
4. (Opcional) Busqueu informació sobre l'atribut `pattern` i limiteu els formats possibles de *Nom d'usuari* i/o *Password*.

TASCA 5 Dissenyeu un nou fitxer HTML que contingui aquest formulari i afegiu l'enllaç a la vostra pàgina principal.

Referències

[W3S23a] W3Schools. *AJAX*. 2023. URL: https://www.w3schools.com/js/js_ajax_intro.asp (consultat 23 de març de 2023).

[W3S23b] W3Schools. *DOM*. 2023. URL: https://www.w3schools.com/js/js_htmldom.asp (consultat 23 de març de 2023).

[W3S23c] W3Schools. *HTML*. 2023. URL: https://www.w3schools.com/html/ (consultat 21 de març de 2023).

[W3S23d] W3Schools. *JS variables*. 2023. URL: https://www.w3schools.com/js/js_variables.asp (consultat 23 de març de 2023).

[Web23] Web Hypertext Application Technology Working Group. *WHATWG*. 2023. URL: https://whatwg.org/ (consultat 21 d'abr. de 2023).

[Wik23] Wikipedia. *HTML*. 2023. URL: https://en.wikipedia.org/wiki/HTML (consultat 21 de març de 2023).